日本国特許庁
JAPAN PATENT OFFICE

別紙添付の書類に記載されている事項は下記の出願書類に記載されている事項と同一であることを証明する。

This is to certify that the annexed is a true copy of the following application as filed with this Office

出願年月日
Date of Application: ２００１年　１月１１日

出願番号
Application Number: 特願２００１－００４１８９

出願人
Applicant(s): インターナショナル・ビジネス・マシーンズ・コーポレーション

CERTIFIED COPY OF
PRIORITY DOCUMENT

２００１年　６月２０日

特許庁長官
Commissioner,
Japan Patent Office 及川耕造

出証番号　出証特２００１－３０５８４４０

【書類名】　特許願
【整理番号】　JP9000230
【提出日】　平成13年 1月11日
【あて先】　特許庁長官　殿
【国際特許分類】　G06F 17/30
【発明者】
【住所又は居所】　神奈川県大和市下鶴間１６２３番地１４　日本アイ・ビー・エム株式会社　東京基礎研究所内
【氏名】　渋谷　哲朗
【特許出願人】
【識別番号】　390009531
【氏名又は名称】　インターナショナル・ビジネス・マシーンズ・コーポレーション
【代理人】
【識別番号】　100086243
【弁理士】
【氏名又は名称】　坂口　博
【代理人】
【識別番号】　100091568
【弁理士】
【氏名又は名称】　市位　嘉宏
【代理人】
【識別番号】　100106699
【弁理士】
【氏名又は名称】　渡部　弘道
【復代理人】
【識別番号】　100104880
【弁理士】
【氏名又は名称】　古部　次郎

【選任した復代理人】

【識別番号】　100100077

【弁理士】

【氏名又は名称】　大場　充

【手数料の表示】

【予納台帳番号】　081504

【納付金額】　21,000円

【提出物件の目録】

【物件名】　明細書　1

【物件名】　図面　1

【物件名】　要約書　1

【包括委任状番号】　9706050

【包括委任状番号】　9704733

【包括委任状番号】　0004480

【プルーフの要否】　要

【書類名】　明細書

【発明の名称】　パターン検索方法、パターン検索装置、コンピュータプログラム及び記憶媒体

【特許請求の範囲】

【請求項１】　検索対象である文字列中から所望のパターンを検索するパターン検索方法において、

前記パターンの最後の文字から前方へ１文字ずつ順に加えて得られる各中間パターンに関して、当該中間パターンの先頭の文字が前記文字列に対する接尾辞配列のどの範囲に存在するかを順次検索する範囲検索ステップと、

前記検索により前記パターン自体に関して得られた前記接尾辞配列の前記範囲に含まれる各要素に対応する前記文字列の要素を特定し、当該文字列の各要素を先頭として前記パターンの要素数と同じ数の要素からなる部分文字列を抽出する文字列抽出ステップと

を含むことを特徴とするパターン検索方法。

【請求項２】　前記範囲検索ステップは、

前記文字列に対する接尾辞配列の各要素に関して、当該各要素に対応する前記文字列中の各文字の一つ前に位置する前置文字を特定するステップと、

前記接尾辞配列中の所定の要素以前の各要素における前記前置文字の中に含まれる、前記パターン中の所望の文字の個数を求めるステップと、

求められた前記文字の個数に基づいて、当該文字が前記接尾辞配列のどの位置に存在するかを検出するステップと

を含むことを特徴とする請求項１に記載のパターン検索方法。

【請求項３】　検索対象である配列中から所望のパターンを検索するパターン検索方法において、

前記パターンの最後の要素が前記配列中のどこに位置するかを検索するステップと、

前記パターンが複数の要素により構成されている場合に、当該パターン中の前記最後の要素に当該最後の要素の前に位置する要素を後ろから順に一つずつ加えて得られる各中間パターンに関して、当該中間パターンが前記配列中のどこに位

置するかを順次検索するステップと
を含むことを特徴とするパターン検索方法。

【請求項４】　検索対象である文字列中から所望のパターンを検索するパターン検索方法において、

前記パターンの最初の文字から後方へ１文字ずつ順に加えて得られる各中間パターンに関して、当該中間パターンの最後の文字が前記文字列に対する接頭辞配列のどの範囲に存在するかを順次検索する範囲検索ステップと、

前記検索により前記パターン自体に関して得られた前記接頭辞配列の前記範囲に含まれる各要素に対応する前記文字列の要素を特定し、当該文字列の各要素を最後尾として前記パターンの要素数と同じ数の要素からなる部分文字列を抽出する文字列抽出ステップと
を含むことを特徴とするパターン検索方法。

【請求項５】　検索対象である遺伝子配列中から所望のパターンを検索するパターン検索方法において、

前記パターンの最後の要素から前方へ一つずつ順に加えて得られる各中間パターンに関して、当該中間パターンの先頭の要素が前記遺伝子配列に対する接尾辞配列のどの範囲に存在するかを順次検索する範囲検索ステップと、

前記検索により前記パターン自体に関して得られた前記接尾辞配列の前記範囲に含まれる各要素に対応する前記遺伝子配列の要素を特定し、当該遺伝子配列の各要素を先頭として前記パターンの要素数と同じ数の要素からなる部分配列を抽出する配列抽出ステップと
を含むことを特徴とするパターン検索方法。

【請求項６】　検索対象である文字列中から所望のパターンを検索するパターン検索装置において、

前記文字列の接尾辞配列に基づいて前記パターンを検索するためのデータ構造を構築する前処理部と、

前記前処理部により構築されたデータ構造を用いて所望の前記パターンを検索する検索部とを備え、

前記前処理部は、

前記接尾辞配列の各要素に関して、当該各要素に対応する前記文字列中の各文字の一つ前に位置する前置文字を特定し、

前記接尾辞配列中の所定の要素以前の各要素における前記前置文字の、前記文字列を構成する文字の種類ごとの個数を求めることにより前記データ構造を構築することを特徴とするパターン検索装置。

【請求項７】　前記前処理部は、前記前置文字の個数を、前記接尾辞配列における要素の位置と、前記文字列を構成する文字の種類とに対応付けて格納したテーブルを持つことを特徴とする請求項６に記載のパターン検索装置。

【請求項８】　前記前処理部は、前記接尾辞配列の所定数個おきの要素の位置に関して生成された前記テーブルを持つことを特徴とする請求項７に記載のパターン検索装置。

【請求項９】　前記前処理部は、前記テーブルにおいて情報が管理される前記接尾辞配列の所定の位置に基づいて、当該位置の間における前記接尾辞配列の要素に対する前記前置文字に関する情報を格納した他のテーブルをさらに持つことを特徴とする請求項８に記載のパターン検索装置。

【請求項１０】　前記検索部は、

前記接尾辞配列中の所定の要素以前の各要素における前記前置文字の個数に基づいて、前記パターンの最後の文字から前方へ１文字ずつ順に加えて得られる各中間パターンに関して、当該中間パターンの先頭の文字が前記文字列に対する接尾辞配列のどの範囲に存在するかを順次検索し、

前記検索により前記パターン自体に関して得られた前記接尾辞配列の前記範囲に含まれる各要素に対応する前記文字列の要素を特定し、当該文字列の各要素を先頭として前記パターンの要素数と同じ数の要素からなる部分文字列を抽出することを特徴とする請求項６に記載のパターン検索装置。

【請求項１１】　コンピュータに、検索対象である配列中から所望のパターンを検索する処理を実行させるコンピュータプログラムにおいて、

前記パターンの最後の要素から前方へ一つずつ順に加えて得られる各中間パターンに関して、当該中間パターンの先頭の要素が前記配列に対する接尾辞配列のどの範囲に存在するかを順次検索する処理と、

前記検索により前記パターン自体に関して得られた前記接尾辞配列の前記範囲に含まれる各要素に対応する前記配列の要素を特定し、当該配列の各要素を先頭として前記パターンの要素数と同じ数の要素からなる部分配列を抽出する処理とを前記コンピュータに実行させることを特徴とするコンピュータプログラム。

【請求項１２】　コンピュータに、検索対象である配列中から所望のパターンを検索する処理を実行させるコンピュータプログラムにおいて、

前記検索対象である配列に対する接尾辞配列の各要素に関して、当該各要素に対応する前記配列中の各要素の一つ前に位置する前置要素を特定する処理と、

前記接尾辞配列中の所定の要素以前の各要素における前記前置要素の、前記配列を構成する要素の種類ごとの個数を求める処理と、

前記接尾辞配列中の所定の要素以前の各要素における前記前置要素の個数に基づいて、前記パターンの最後の要素から前方へ一つずつ順に加えて得られる各中間パターンに関して、当該中間パターンの先頭の要素が前記配列に対する接尾辞配列のどの範囲に存在するかを順次検索する処理と、

前記検索により前記パターン自体に関して得られた前記接尾辞配列の前記範囲に含まれる各要素に対応する前記配列の要素を特定し、当該配列の各要素を先頭として前記パターンの要素数と同じ数の要素からなる部分配列を抽出する処理とを前記コンピュータに実行させることを特徴とするコンピュータプログラム。

【請求項１３】　前記前置要素の個数を、前記接尾辞配列における要素の位置と、前記配列を構成する要素の種類とに対応付けて格納したテーブルを生成し、記憶装置に保持する処理を前記コンピュータにさらに実行させることを特徴とする請求項１２に記載のコンピュータプログラム。

【請求項１４】　コンピュータに検索対象である配列中から所望のパターンを検索する処理を実行させるプログラムを当該コンピュータの入力手段が読取可能に記憶した記憶媒体において、

前記プログラムは、前記パターンを検索する処理として、

前記パターンの最後の要素から前方へ一つずつ順に加えて得られる各中間パターンに関して、当該中間パターンの先頭の要素が前記配列に対する接尾辞配列のどの範囲に存在するかを順次検索する処理と

前記検索により前記パターン自体に関して得られた前記接尾辞配列の前記範囲に含まれる各要素に対応する前記配列の要素を特定し、当該配列の各要素を先頭として前記パターンの要素数と同じ数の要素からなる部分配列を抽出する処理とを前記コンピュータに実行させることを特徴とする記憶媒体。

【請求項１５】　コンピュータに検索対象である配列中から所望のパターンを検索する実行させるプログラムを当該コンピュータの入力手段が読取可能に記憶した記憶媒体において、

前記プログラムは、前記パターンを検索する処理として、

前記接尾辞配列の各要素に関して、当該各要素に対応する前記配列中の各要素の一つ前に位置する前置要素を特定する処理と、

前記接尾辞配列中の所定の要素以前の各要素における前記前置要素の、前記配列を構成する要素の種類ごとの個数を求める処理と、

前記接尾辞配列中の所定の要素以前の各要素における前記前置要素の個数に基づいて、前記パターンの最後の要素から前方へ一つずつ順に加えて得られる各中間パターンに関して、当該中間パターンの先頭の要素が前記配列に対する接尾辞配列のどの範囲に存在するかを順次検索する処理と、

前記検索により前記パターン自体に関して得られた前記接尾辞配列の前記範囲に含まれる各要素に対応する前記配列の要素を特定し、当該配列の各要素を先頭として前記パターンの要素数と同じ数の要素からなる部分配列を抽出する処理とを前記コンピュータに実行させることを特徴とする記憶媒体。

【発明の詳細な説明】

【０００１】

【発明の属する技術分野】

本発明は、文字列などの配列中に存在する頻出部分配列や、二つ以上の配列に共通な部分配列を検索するためのデータ構造及びこのデータ構造を用いたパターン検索方法に関する。

【０００２】

【従来の技術】

文字列中に存在する頻出部分文字列や、二つ以上の文字列に共通な部分文字列

などを高速で検索するのに有効なデータ構造として、接尾辞木（Suffix tree）が知られている。接尾辞木は、処理対象の文字列中に存在しない文字＄を処理対象の文字列の最後に加えた文字列における全ての接尾辞を表す木である。接尾辞木の葉ノード（各枝において他の枝が接続されていない先端のノード）は、それぞれの接尾辞に対応する。

ここで、接尾辞とは、所定の文字列において、所定の文字を特定した場合の当該文字以降の文字列である。

図６は、接尾辞木の例を示す図である。図６には、処理対象の文字列として「ｍｉｓｓｉｓｓｉｐｐｉ」の最後に文字＄を加えた文字列「ｍｉｓｓｉｓｓｉｐｐｉ＄」の接尾辞木を示す。

【０００３】

接尾辞木において、各枝は、部分文字列に相当するラベルを持つ。そして、ルートノードから葉ノードまでの各枝が持つラベルを並べたものが、当該葉ノードに対応する接尾辞となる。図６に示す例では、例えば、ルートノードから「ｉ」「ｓｓｉ」「ｐｐｉ」のラベルを持つ各枝を経て到達する葉ノードに対応する接尾辞は「ｉｓｓｉｐｐｉ」であり、同様に「ｓ」「ｓｉ」「ｓｓｉｐｐｉ」のラベルを持つ各枝を経て到達する葉ノードに対応する接尾辞は「ｓｓｉｓｓｉｐｐｉ」である。

【０００４】

また、接尾辞木における単一のノード（ルートノードを含む）から出てゆく各枝に付されているラベルの最初の文字は全て異なり、これらはラベルの最初の文字でソートされている。図６に示す例では、図の左側から右側へ向けて英語のアルファベット順（ｉ、ｍ、ｐ、ｓの順）に枝が並んでいる。

【０００５】

接尾辞木を生成するアルゴリズムとしては、処理対象である文字列の長さをｎ、文字列を構成するアルファベットのサイズ（文字の種類の数）をｓとした場合、Ｏ（ｎ log ｓ）のアルゴリズムが知られている。特にアルファベットが整数アルファベット（１からｎまでの数字）である場合は、Ｏ（ｎ）のアルゴリズムが知られている。ここで、Ｏ（func(n)）は、実際の計算時間がｔである場合に

、ｎ≧ｋであるようなｎに対して、

$$0 \leqq t \leqq c \times func(n)$$

が成り立つような何らかの定数ｃとｋの組が必ず存在することを意味する。したがって、Ｏ（ｎ ｌｏｇ ｓ）はｎ ｌｏｇ ｓの定数倍以内の時間で計算が可能であることを意味し、Ｏ（ｎ）はｎの定数倍以内の時間（この場合、ｎも定数なので、定数時間内）で計算できることを意味する。

【０００６】

これを用いれば、長さｍの部分文字列の検索は、Ｏ（ｍ ｌｏｇ ｓ）に相当する時間で行うことができる。通常、アルファベットのサイズは定数サイズなので、この時間は線形時間といって良い。

英文字テキスト（ｎ文字）に対するこの接尾辞木を扱うために必要とする記憶装置の記憶容量は、２０ｎバイト～４０ｎバイトである。

【０００７】

この接尾辞木のデータサイズは大きいため、このデータサイズを抑制する類似のパターン検索用のデータ構造として、接尾辞配列（suffix array）が知られている。

上述したように、接尾辞木の葉ノードは、それぞれが文字列の接尾辞に対応している。この接尾辞を、接尾辞木の一端側（図６の例では左端側）の葉ノードに対応する接尾辞から順に並べると、処理対象の文字列における全ての接尾辞を辞書的順序で並べた配列が得られる。ただし、各接尾辞は、最後に終了判定文字＄を付加されているものとする。

【０００８】

この配列の構成要素である各接尾辞を、処理対象の文字列における当該接尾辞の最初の文字の位置を表す情報で置き換える（例えば、「ｉｐｐｉ＄」を「８」に、「ｉｓｓｉｐｐｉ＄」を「５」にというように置き換える）。これにより、処理対象の文字列と同じ長さの配列（接尾辞配列）が得られる。例えば、図６における「ｍｉｓｓｉｓｓｉｐｐｉ＄」の接尾辞配列は、「８　５　２　１１　１

１０　９　７　４　６　３　１２」となる。なお、文字＄は他の全ての文字よりも辞書的順序が後であるとしている。

【０００９】

この接尾辞配列を用いると、接尾辞木を用いる場合と比較して、文字列検索を行うために必要なメモリ容量を削減することができる。また、文字列の検索に要する時間は、２分探索を行うことから、Ｏ（ｐ log ｑ）となる。ただし、ｑはデータベースの大きさ、ｐは検索しようとする文字列の長さである。

通常、必要な記憶容量は一つの文字に対し４バイトであるから、テキストが英文字（１バイト）の場合、ｎ文字のテキストに対するこのデータベースのデータサイズは５ｎバイトである。

【００１０】

また、データベースに、さらに隣接する接尾辞の共通接頭辞長のテーブルを持たせることもできる。このテーブルを用いると、接尾辞木配列のみを用いる場合に対して、検索時間をＯ（ｐ＋log ｑ）と短縮することができる。なお、この場合におけるデータベースのデータサイズは９ｎバイトとなる。

【００１１】

【発明が解決しようとする課題】

大規模なテキストデータベースを検索するために、上述した接尾辞木や接尾辞配列をデータ構造として用いる場合、次のような問題がある。

まず、接尾辞木をデータ構造として用いる場合、必要とされるデータベースの大きさが大きいという問題がある。

上述したように、処理対象である文字列の長さがｎである場合、このテキストに対する接尾辞木を扱うために必要な記憶装置の記憶容量、すなわちデータベースのデータサイズは、２０ｎバイト～４０ｎバイトである。一般に、データ構造として接尾辞木を用いる場合、記憶装置に対して、接尾辞配列を用いる場合の４～６倍の記憶領域（接尾辞配列では、１バイト文字で文字数ｎのテキストの場合、５ｎバイト）を要する。

このため、大規模なテキストデータベースに対して接尾辞木を使用することは困難である。

【００１２】

一方、接尾辞配列をデータ構造として用いる場合、検索に長時間を要するという問題がある。

接尾辞配列に対して検索を行う場合、２分探索を行うため、データベースの大きさをｑ、検索しようとする文字列の長さをｐとして、Ｏ（ｐ log ｑ）だけの時間を要する。したがって、アルファベットのサイズが定数サイズである場合にほぼ線形時間で探索を行うことができる接尾辞木に比べて、多大な計算時間を要する。

また上述したように、データサイズが多少大きくなることを許し、データベースに、接尾辞配列中で隣接する接尾辞の共通接頭辞長のテーブルを持たせることによって、計算時間をＯ（ｐ＋log ｑ）に短縮することができる。しかしこの場合であっても、依然としてlog ｑの項が残っているため、接尾辞木の場合と比べると、多大な計算時間を要する。

【００１３】

そこで本発明は、大規模テキストデータベースの検索において、処理を行うためのデータ構造におけるデータサイズの増大を抑えながら、高速な検索を実現することを目的とする。

【００１４】

【課題を解決するための手段】

かかる目的のもと、本発明は、検索対象である文字列中から所望のパターンを検索するパターン検索方法において、次の範囲検索ステップと、文字列抽出ステップとを含むことを特徴とする。すなわち、範囲検索ステップにおいて、このパターンの最後の文字から前方へ１文字ずつ順に加えて得られる各中間パターンに関して、この中間パターンの先頭の文字が検索対象の文字列に対する接尾辞配列のどの範囲に存在するかを順次検索する。この検索をパターンの最後の文字から順に実行することによって、最終的に、このパターン自体を含む接尾辞配列の範囲が求められる。次に、文字列抽出ステップにおいて、当該接尾辞配列の範囲に含まれる各要素に対応する文字列の要素を特定し、この文字列の各要素を先頭としてこのパターンの要素数と同じ数の要素からなる部分文字列を抽出する文字列

抽出ステップとを含むことを特徴とする。

上記のように構成されたパターン検索は、アルファベットや日本語のテキストなど種々の文字による文字列における検索に用いることができるが、バイナリデータや遺伝子配列のような使用される文字の種類が少ない文字列から所望のパターンを検索する場合には、検索に用いるデータ構造のデータサイズを特に小さくすることができる。

【００１５】

ここで、この範囲検索ステップは、検索対象の文字列に対する接尾辞配列の各要素に関して、この各要素に対応する文字列中の各文字の一つ前に位置する前置文字を特定するステップと、接尾辞配列中の所定の要素以前の各要素における前置文字の中に含まれる、このパターン中の所望の文字の個数を求めるステップと、求められた文字の個数に基づいて、この文字が接尾辞配列のどの位置に存在するかを検出するステップとを含むことを特徴とする。

【００１６】

また、本発明は、検索対象である配列中から所望のパターンを検索するパターン検索方法において、このパターンの最後の要素が前記配列中のどこに位置するかを検索するステップと、このパターンが複数の要素により構成されている場合に、このパターン中の最後の要素に、この最後の要素の前に位置する要素を後ろから順に一つずつ加えて各中間パターンを得、この中間パターンが配列中のどこに位置するかを順次検索するステップとを含むことを特徴とする。

【００１７】

また、本発明は、上記のように接尾辞配列を用いるパターン検索だけではなく、接頭辞配列を用いるパターン検索にも適用することができる。すなわち、上述した範囲検索ステップにおいて、このパターンの最初の文字から後方へ１文字ずつ順に加えて得られる各中間パターンに関して、この中間パターンの最後の文字が検索対象の文字列に対する接頭辞配列のどの範囲に存在するかを順次検索する。この検索をパターンの最初の文字から順に実行することによって、最終的に、このパターン自体を含む接頭辞配列の範囲が求められる。次に、文字列抽出ステップにおいて、当該接頭辞配列の範囲に含まれる各要素に対応する文字列の要素

を特定し、この文字列の各要素を最後尾としてこのパターンの要素数と同じ数の要素からなる部分文字列を抽出する文字列抽出ステップとを含む構成とすることができる。

【００１８】

また、本発明は、検索対象である文字列中から所望のパターンを検索するパターン検索装置において、この文字列の接尾辞配列に基づいてパターンを検索するためのデータ構造を構築する前処理部と、この前処理部により構築されたデータ構造を用いて所望のパターンを検索する検索部とを備え、この前処理部は、接尾辞配列の各要素に関して、この各要素に対応する文字列中の各文字の一つ前に位置する前置文字を特定し、接尾辞配列中の所定の要素以前の各要素における前置文字の、検索対象の文字列を構成する文字の種類ごとの個数を求めることによりデータ構造を構築することを特徴とする。

【００１９】

ここで、前処理部は、前置文字の個数を、接尾辞配列における要素の位置と、検索対象の文字列を構成する文字の種類とに対応付けて格納したテーブルを持つことができる。

ここでさらに、このテーブルを、接尾辞配列の所定数個おきの要素の位置に関するテーブルとすることができる。すなわち、テーブルに格納するデータを間引くことにより、当該データ構造のデータサイズを縮小することができる。

さらにこの場合、前処理部は、間引いた範囲の前置文字の個数を算出する際に使用するため、このテーブルにおいて情報が管理される接尾辞配列の所定の位置に基づいて、この位置の間における接尾辞配列の要素に対する前置文字に関する情報を格納した他のテーブルをさらに持つことができる。

【００２０】

また、このパターン検索装置において、検索部は、この接尾辞配列中の所定の要素以前の各要素における前置文字の個数に基づいて、パターンの最後の文字から前方へ１文字ずつ順に加えて得られる各中間パターンに関して、この中間パターンの先頭の文字が文字列に対する接尾辞配列のどの範囲に存在するかを順次検索する。この検索により、このパターン自体を含む接尾辞配列の範囲が得られる

。そして、この範囲に含まれる各要素に対応する文字列の要素を特定し、この文字列の各要素を先頭としてパターンの要素数と同じ数の要素からなる部分文字列を抽出する。

【００２１】

また、本発明は、コンピュータに、検索対象である配列中から所望のパターンを検索する処理を実行させるコンピュータプログラムにおいて、このパターンの最後の要素から前方へ一つずつ順に加えて得られる各中間パターンに関して、この中間パターンの先頭の要素が検索対象である配列に対する接尾辞配列のどの範囲に存在するかを順次検索する処理と、この検索によりこのパターン自体に関して得られた接尾辞配列の範囲に含まれる各要素に対応する配列の要素を特定し、この配列の各要素を先頭としてこのパターンの要素数と同じ数の要素からなる部分配列を抽出する処理とをコンピュータに実行させることを特徴とする。

【００２２】

さらにまた、本発明は、コンピュータに、検索対象である配列中から所望のパターンを検索する処理を実行させるコンピュータプログラムにおいて、検索対象である配列に対する接尾辞配列の各要素に関して、この各要素に対応する配列中の各文字の一つ前に位置する要素を特定する処理と、この接尾辞配列中の所定の要素以前の各要素における前置要素の、配列を構成する要素の種類ごとの個数を求める処理と、この接尾辞配列中の所定の要素以前の各要素における前置要素の個数に基づいて、このパターンの最後の要素から前方へ１文字ずつ順に加えて得られる各中間パターンに関して、この中間パターンの先頭の要素がこの配列に対する接尾辞配列のどの範囲に存在するかを順次検索する処理と、この検索によりこのパターン自体に関して得られた接尾辞配列の範囲に含まれる各要素に対応する配列の要素を特定し、この配列の各要素を先頭としてこのパターンの要素数と同じ数の要素からなる部分配列を抽出する処理とをコンピュータに実行させることを特徴とする。

これらのコンピュータプログラムは、例えば磁気ディスクその他の記憶媒体に格納して提供することができる。また、インターネットなどのネットワークを介して伝送させることにより提供することもできる。

【００２３】

【発明の実施の形態】

以下、添付図面に示す実施の形態に基づいてこの発明を詳細に説明する。

図１は、本実施の形態を実現するのに好適なコンピュータ装置のハードウェア構成の例を模式的に示した図である。

図１に示すコンピュータ装置は、ＣＰＵ（中央処理装置）１０１と、システムバスを介してＣＰＵ１０１に接続されたＭ／Ｂ（マザーボード）チップセット１０２及びメインメモリ１０３と、ＰＣＩバスなどの高速なバスを介してＭ／Ｂチップセット１０２に接続されたビデオカード１０４、ハードディスク１０５及びネットワークインタフェース１０６と、さらにブリッジ回路１１０及びＩＳＡバスなどの低速なバスを介してＭ／Ｂチップセット１０２に接続されたフロッピーディスクドライブ１０７、キーボード１０８及びシリアルＩ／Ｏポート１０９とを備える。

なお、図１は本実施の形態による検索方法を実現するコンピュータ装置の構成を例示するに過ぎず、本実施の形態を適用可能であれば、他の種々のシステム構成を取ることが可能である。

【００２４】

本実施の形態は、図１に示したメインメモリ１０３に展開されたプログラムにてＣＰＵ１０１を制御することにより、所定の文字列（文字を要素とする配列）中から所望の部分文字列を検索する（以下、検索対象の文字列をテキスト、検索する部分文字列をパターンと称す）。

図２は、プログラム制御されたＣＰＵ１０１において、本実施の形態におけるデータ構造の構築及び検索を行うための機能ブロックを示す図である。

図２を参照すると、本実施の形態は、テキストの接尾辞配列を生成する接尾辞配列生成部１０と、接尾辞配列生成部１０にて生成された接尾辞配列を変換して所望のデータ構造を構築する前処理部２０と、前処理部２０にて構築されたデータ構造を用いてパターンの検索を行う検索部３０とを備える。

【００２５】

上述したように、これらの構成要素は、プログラム制御されたＣＰＵ１０１に

より実現される仮想的なソフトウェアブロックである。当該プログラムは、磁気ディスクや光ディスク、半導体メモリ、その他の記憶媒体に格納して提供したり、ネットワークを介して伝送したりすることができる。本実施の形態は、図１に示したネットワークインタフェース１０６やフロッピーディスクドライブ１０７、図示しないＣＤ－ＲＯＭドライブなどを介して当該プログラムを入力し、ハードディスク１０５に格納する。そして、ハードディスク１０５に格納されたプログラムをメインメモリ１０３に読み込んで展開し、ＣＰＵ１０１にて実行する。

【００２６】

図２において、接尾辞配列生成部１０は、図示しないデータベースから検索対象であるテキストを取得し、接尾辞配列を生成する。接尾辞配列の生成方法としては、公知の任意のアルゴリズムを用いることができる。検索対象であるテキストや生成された接尾辞配列は、メインメモリ１０３に格納される。なお、接尾辞配列は、公知の種々の方法により生成することが可能であり、外部装置において生成された接尾辞配列を本実施の形態において使用することもできる。したがって、接尾辞配列生成部１０は必須の構成要素ではない。接尾辞配列生成部１０を構成要素として設けない場合は、検索対象となるテキストと当該テキストの接尾辞配列とがメインメモリ１０３に直接格納されることとなる。

【００２７】

以下の説明において、検索対象となるテキストをＴ［１・・・ｎ］、検索するパターンをＰ［１・・・ｍ］とする。また、テキストＴに対する接尾辞配列をＳＡ［１・・・ｎ］とする。なお、以下の説明では、テキストＴやパターンＰの文字列はダブルクオーテーションマーク（“　”）で囲み、その中の文字はクオーテーションマーク（‘　’）で囲んで示すこととする。

例えば、「ｍｉｓｓｉｓｓｉｐｐｉ」の最後に文字＄を加えたテキストＴは、

Ｔ［１・・・１２］＝“ｍｉｓｓｉｓｓｉｐｐｉ＄”

となる。ここで、‘＄’は、終了判定文字であり、辞書的順序が他の全ての文字よりも大きい（すなわち後に位置する）ものとする。また、テキストＴ［１・・

・１２］から３文字のパターン「ｓｓｉ」を検索する場合、検索パターンＰは、

Ｐ［１・・・３］＝“ｓｓｉ”

となる。さらにここで、テキストＴに対し、Ｔ［５］＝‘ｉ’（５番目の文字）のように表すと、テキストＴに対する接尾辞配列ＳＡは、

ＳＡ［１・・・１２］
＝｛８　５　２　１１　１　１０　９　７　４　６　３　１２｝

となる。

なお、終了判定文字‘$’は、概念上のものとして扱い、実際の処理においてはメインメモリ１０３に格納しなくても良い。この場合、Ｔ［ｉ］にアクセスする際、ｉ＝１２ならば‘$’である、という分岐条件を入れることとなる。例えば、テキストＴの文字数が２５６個あり、‘$’も１文字として扱うと１バイトに収まらなくなってしまうような場合は、メインメモリ１０３に格納しない方が望ましい。

【００２８】

前処理部２０は、接尾辞配列生成部１０により生成された接尾辞配列ＳＡを読込み、これに基づいて、検索対象である文字列からｆ（ｉ，ｃ）（ｉはｎ以下の正の整数、ｃは文字）で定義されるデータを検出するためのデータ構造を構築する。

ここで、ｆ（ｉ，ｃ）は、Ｔ［ＳＡ［ｊ］－１］＝ｃ（ｊ≦ｉ）であるようなｊの数である。配列Ｂを考え、Ｂ［ｉ］＝Ｔ［ＳＡ［ｉ］－１］とする。すなわち、配列Ｂは、接尾辞配列ＳＡの各要素に対応するテキストＴの各文字の一つ前に位置する文字（前置文字）の配列である。例えば、Ｂ［４］＝Ｔ［ＳＡ［４］－１］＝Ｔ［１０］＝‘ｉ’となる。同様にして配列Ｂの全ての文字を書き出すと次のようになる。

Ｂ［１・・・１２］＝“ｓｓｍｐ＄ｐｉｓｓｉｉｉ”

したがって、上記のｆ（ｉ，ｃ）の値は、配列Ｂにおいて、インデックスがｉ以下での文字ｃの個数で表現することができる。例えば、ｆ（６，‘ｓ’）＝２であり、ｆ（６，‘ｐ’）＝２であり、ｆ（６，‘ｍ’）＝１である。なお、ｉ＞ｎであるようなｉに対しては、ｆ（ｉ，ｃ）＝ｆ（ｎ，ｃ）と定義する。また、ｉ≦０であるようなｉに対しては、ｆ（ｉ，ｃ）＝０と定義する。

【００２９】

ｆ（ｉ，ｃ）のデータ全体をテーブルとして保持すれば、パラメータであるｉ、ｃを与えれば直ちに対応するｆ（ｉ，ｃ）を求めることができる。しかし、テキストＴを構成する文字の種類（ｓ）が極めて少ない場合、例えばバイナリデータの文字列（２種類：０、１）やＤＮＡ配列（４種類：アデニン（Ａ）、チミン（Ｔ）、グアニン（Ｇ）、シトシン（Ｃ））では可能であるが、文字の種類（ｓ）が多い場合は、当該テーブルは極めて大きな配列となるため、現実的ではない。

そこで、前処理部２０は、以下のようにしてｆ（ｉ，ｃ）を算出するためのデータ構造を構築する。

【００３０】

（１）テーブルＦの作成

ｋを適当な大きさのｎ以下の正の整数であるとする。まず、すべての正の整数ｉ（ｋ＊ｉ＜ｎ＋ｋ）に対して、ｆ（ｋ＊ｉ，ｃ）のテーブルを作成する。これは、テキストＴをｋ個の文字ごとに区切り、ｋ番目の文字ごとにｆ（ｉ，ｃ）を求めてテーブルを作成することに相当する。ｆ（）の大きさはｎ以下であるから、このサイズは（ｎ＊ｓ log ｎ）／ｋビットである。これは、ｎが１ワードに入る通常のケースではＯ（ｎ＊ｓ／ｋ）ワード（すなわち、ｎ＊ｓ／ｋの定数倍以内）のことである。

このテーブルをＦとし、

Ｆ［ｉ］［ｃ］＝ｆ（ｋ＊ｉ，ｃ）

とする。なお、このテーブルＦは、テーブルの大きさとテキストの大きさのうち大きい方に比例した時間で構築することができる。

【００３１】

このテーブルＦを持つことにより、ｋの倍数のインデックスに関しては、ｆをＯ（１）の時間で求めることができる。そこで次に、ｋの倍数以外のインデックスに関してｆの値を求めるためのデータ構造を考える。

そのため、ｇ（ｉ，ｃ，ｊ）を、Ｔ［ＳＡ［ｐ］－１］＝ｃを満たすｐ（ただし、ｋ＊（ｉ－１）＜ｐ≦ｋ＊ｉ）のうち、ｊ番目のものとし、まず、このｇ（ｉ，ｃ，ｊ）を求めるためのデータ構造について述べる。

【００３２】

（２）テーブルＬの作成

ｈ（ｉ，ｃ）を、ｆ（ｋ＊ｉ，ｃ）－ｆ（ｋ＊（ｉ－１），ｃ）とする。これはテーブルＦから直ちに計算可能である。

ｌ（ｉ，ｃ）を、ｈ（ｉ，ｄ）（ｄ＜ｃ，辞書順）の総和として、これをテーブルとして持つ。このテーブルをＬとし、

Ｌ［ｉ］［ｃ］＝ｌ（ｉ，ｃ）

とする。このテーブルＬのサイズは（ｎ＊ｓ log ｋ））／ｋビットである。

【００３３】

（３）テーブルＧの作成

次に、全てのｒ（０＜ｋ＊ｒ＜ｎ＋ｋ）に対して、０＜ｑ≦ｋを満たす整数ｑを、Ｔ［ＳＡ［ｑ＋ｋ＊（ｒ－１）］－１］の値が同じ物ごとに辞書的順序にしたがって並べ替えたものをテーブルＧ［ｒ］［１・・・ｋ］とする。このとき、Ｔ［ＳＡ［ｑ＋ｋ＊（ｒ－１）］－１］の値における辞書的順序が同じものに関してはｑの値が小さいものが先になるように並べる。ただし、

【数１】

$$r=\lceil n/k \rceil$$

の場合、０＜ｑ≦ｎ－（ｒ－１）ｋのようなｑだけを並べる。これは、数１を満足するｒの範囲に含まれる文字の数がｋ個に満たない場合があるためである。すなわち、上述したようにテーブルＦの作成において、テキストＴをｋ個の文字ごとに区切ったが、テキストＴの文字数ｎがｋで割り切れない場合は、最後尾の区分における文字数はｋ個に満たない。したがって、０＜ｑ≦ｎ－（ｒ－１）ｋのようなｑを並べることとする。

テーブルＧの配列のサイズは全体でｎであり、ビットで表すとｎ log ｋビットということになる。これは、例えば次のようにして求めることができる。

```
for allc (すべての文字) w(c)=1;
for (q= 1;q<=k;q++) {
  c = T[SA[q+k*(r-1)]-1];
  G[r][L[r,c]+w(c)]= q;
  w(c) = w(c)+1;
}
```

ただし、ｒの値が上記数１を満足する場合、forループは（ｑ＝１；ｑ≦ｎ－（ｒ－１）ｋ；ｑ＋＋）となる。

【００３４】

（４）ｆ（ｉ，ｃ）の計算

上記のようにして作成されたテーブルＧ及びテーブルＬを用いて、ｇ（ｉ，ｃ，ｊ）を示すと、

ｇ（ｉ，ｃ，ｊ）＝Ｇ［ｉ］［Ｌ［ｉ］［ｃ］＋ｊ］＋ｋ＊（ｉ－１）

である。したがって、ｇ（ｉ，ｃ，ｊ）はＧ、Ｌの二つテーブルからＯ（１）時間で得ることができる。

次に、ｋ＊（ｊ－１）＜ｊ≦ｋ＊ｉであるようなｊに対し、Ｔ［ＳＡ［ｐ］－１］＝ｃ（ただしｋ＊（ｉ－１）＜ｐ≦ｊ）となるようなｐの数をｆ’（ｊ，ｃ）とする。そして、ｘ（０＜ｘ≦ｈ（ｉ，ｃ））の区間で、ｇ（ｉ，ｃ，ｘ）の値がｊ以下となるような最大のｘを見つけ出すと、

ｆ’（ｊ，ｃ）＝ｘ

となる。このｘの値は、ｇ（ｉ，ｃ，ｘ）の値が昇順になっているため、２分探索によりＯ（ log ｈ（ｉ，ｃ））で計算可能である。ｈ（ｉ，ｃ）＜ｋであるから、これは、Ｏ（ log ｋ）ということである（ｈ（ｉ，ｃ）の平均値はｋ／ｓであるため、実際にはより短い時間で計算できる）。ただし、このようなｘが存在しない場合は、ｆ’（ｊ，ｃ）＝０とする。

以上の前提で、ｆ（ｊ，ｃ）は、

ｆ（ｊ，ｃ）＝Ｆ［ｉ－１］［ｃ］＋ｆ’（ｊ，ｃ）

と計算できる。したがって、ｆ（ｊ，ｃ）は、以上のデータ構造を用いることにより、Ｏ（ log ｋ）で計算することができる。

【００３５】

上述したテーブルＦ、Ｌ、Ｇを表すのに必要なビット数は、テーブルＦが（ｎ＊ｓ log ｎ）／ｋビット、テーブルＬが（ｎ＊ｓ log ｋ））／ｋビット、テーブルＧがｎ log ｋビットであるから、全体で

（ｎ＊ｓ／ｋ）＊（log ｎ＋log ｋ）＋ｎ log ｋビット

である。これらのテーブルＦ、Ｌ、Ｇは、メインメモリ１０３に格納される。

実際の運用においては、メインメモリ１０３の記憶容量として、これに加えて接尾辞配列ＳＡのためのｎ log ｎビット及びテキストＴ自身のためのｎ log ｓビットが必要になる。

【００３６】

また、前処理部２０は、テキストＴに関して、これらのデータ構造に加えて次に示すテーブルＣも持つこととする。このテーブルの要素Ｃ［ｃ］は、テキストＴに含まれるｃ以下の文字の総数を表す。ただし、ｃ以下の文字とは、ｃあるいはｃより辞書的順序で早い文字を意味する。

テーブルＣも他のデータ構造と同様に、メインメモリ１０３に格納される。このテーブルＣのサイズはｓ log ｎビットである。また、テーブルＣはテキストＴに対し、線形時間で計算可能である。

なお、ｋの値を小さく設定した場合には、ｊ＝ｉ＊ｋ＋ｄ（ｄ＜ｋ）に対して、ｆ（ｉ＊ｋ，ｃ）を求める際、テーブルＬ、Ｇは持たずに、テーブルＦから求められるｆ（ｉ＊ｋ，ｃ）の値と、ｉ＊ｋ＋１とに基づいて、Ｔ［ＳＡ［ｊ］－１］の値がｃであるものの個数を数えるという方法も考えられる。この場合の計算時間はＯ（ｋ）であるので、ｋとlog ｋの値が近いような小さなｋに対しては有効である。この方法を用いる場合は、テーブルＬ、Ｇを持たない分、必要なメモリの記憶容量は減少する。

【００３７】

検索部３０は、前処理部２０にて作成された上記のデータ構造を用いて、テキストＴから所望のパターンＰを検索する。

検索は、ｆ（ｉ，ｃ）を用い、次のように行う。

```
start = C[P[m]-1]+1 ;
end = C[P[m]] ;
for each i (m-1>=i>=1, 降順) {
   c = P[i];
   start = C[c-1] + f(start, c);
```

```
    end = C[c-1] + f(end, c);
    if (end < start) {
        パターンは存在しないので終了。
    }
}
```

ただし、文字ｃに対し、ｃ＋１とは、辞書的順序で文字ｃの次に来る文字を表し、ｃ－１は辞書的順序で文字ｃの前に来る文字を表すものとする。ただし、辞書的順序で最小のアルファベットａに対しては、ｃ［ａ－１］は０を表すものとする。

【００３８】

図３は、上記の検索アルゴリズムに対応するフローチャートである。同図を参照して、本実施の形態によるパターンの検索手順を説明する。この検索方法は、パターンを当該パターンの構成文字列の後ろから検索することが特徴である。図３に示す検索アルゴリズムにより、求めるパターンは、テキストＴの接尾辞配列ＳＡにおいて、ＳＡ［ｊ］（ｓｔａｒｔ≦ｊ≦ｅｎｄ）の位置から始まる場所に存在するので、それを列挙すればよい。

【００３９】

図３を参照すると、検索部３０は、まず、ｓｔａｒｔにＣ［Ｐ［ｍ］－１］＋１を代入し、ｅｎｄにＣ［Ｐ［ｍ］］を代入し、ｓｔａｒｔとｅｎｄの値を求める。また、ｉ＝ｍ－１とする（ステップ３０１）。

次に、ｉの値が正（ｉ＞０）かどうかを調べ、正であれば、次に、ｃ＝Ｐ［ｉ］として、ｓｔａｒｔにＣ［ｃ－１］＋ｆ（ｓｔａｒｔ，ｃ）を代入し、ｅｎｄにＣ［ｃ－１］＋ｆ（ｅｎｄ，ｃ）を代入し、ｓｔａｒｔとｅｎｄの値を求める。また、ｉ＝ｉ－１とする（ステップ３０２、３０３）。

次に、ｅｎｄの値がｓｔａｒｔの値を下回ったかどうかを調べ、下回ったならば、検索パターンＰにマッチする文字列はテキストＴには存在しないことがわかるので、処理を終了する（ステップ３０４、３０５）。

一方、ｅｎｄの値がｓｔａｒｔの値を下回っていなければ、ステップ３０２に

戻って、新たなｉに関してｓｔａｒｔ及びｅｎｄの値を求める（ステップ３０４）。

ステップ３０２において、ｉの値が０以下になったならば、ｓｔａｒｔ及びｅｎｄの値を用い、ｓｔａｒｔ≦ｊ≦ｅｎｄであるような全てのｊに対して、ＳＡ［ｊ］の位置から始まるテキストＴの接尾辞を出力して処理を終了する（ステップ３０６）。このとき、当該接尾辞と検索パターンＰとがマッチする。

【００４０】

次に、

Ｔ［１・・・１２］＝“ｍｉｓｓｉｓｓｉｐｐｉ＄”
Ｐ［１・・・３］＝“ｓｓｉ”
ＳＡ［１・・・１２］
＝｛８　５　２　１１　１　１０　９　７　４　６　３　１２｝

の場合について、前処理部２０によるデータ構造の構築及び検索部３０によるパターンＰの検索の動作例を説明する。

本動作例では、テキストＴを区切る基準としてｋ＝４とする。

【００４１】

まず、ｋ＝４の場合のテーブルＦを作成する。

上述したように、Ｆ［ｉ］［ｃ］にはｆ（ｋ＊ｉ，ｃ）が入る。そして、ｋ＊ｉ＜ｎ＋ｋであり、ｎ＝１２であるから、ｋ＝４の場合、ｉの値は１、２、３である。したがって、テーブルＦには、ｉ＝１、２、３及びｃ＝‘ｉ’、‘ｍ’、‘ｐ’、‘ｓ’、‘＄’の各々について、ｆ（４＊ｉ，ｃ）の値が入り、図４に示すようになる。

例えば、Ｆ［２］［‘ｐ’］の場合、ｆ（４＊２，‘ｐ’）であるから、配列Ｂ［１・・・１２］＝“ｓｓｍｐ＄ｐｉｓｓｉｉｉ”において８（＝４＊２）番目の文字である‘ｓ’以前に‘ｐ’は２個存在する。したがって、テーブルＦのＦ［２］［‘ｐ’］には２が入る。なお、図４のテーブルＦでは、‘＄’に対するエントリーも入れているが、実際には、検索パターンの中に‘＄’が入ること

は考えなくて良いので、‘$’に対する列は省略することができる。

【００４２】

次に、テーブルＬを作成する。

上述したように、ｈ（ｉ，ｃ）をｆ（４＊ｉ，ｃ）－ｆ（４＊（ｉ－１），ｃ）とし、ｌ（ｉ，ｃ）を、ｈ（ｉ，ｄ）（ｄ＜ｃ，辞書順）の総和とすると、文字ｃの順序は‘ｉ’＜‘ｍ’＜‘ｐ’＜‘ｓ’＜‘$’であるから、テーブルＬは、図５に示すようになる。

例えば、Ｌ［２］［‘ｓ’］は、ｈ（２，‘ｉ’）とｈ（２，‘ｍ’）とｈ（２，‘ｐ’）との総和であり、図４のテーブルＦを参照すれば、

ｈ（２，‘ｉ’）＝ｆ（４＊２，‘ｉ’）－ｆ（４＊（２－１），‘ｉ’）
＝ｆ（８，‘ｉ’）－ｆ（４，‘ｉ’）＝１－０＝１

すなわち、配列Ｂ［５・・・８］において、‘ｉ’が一つ存在することがわかる。同様に、

ｈ（２，‘ｍ’）＝ｆ（８，‘ｍ’）－ｆ（４，‘ｍ’）＝１－１＝０
ｈ（２，‘ｐ’）＝ｆ（８，‘ｐ’）－ｆ（４，‘ｐ’）＝２－１＝１

したがって、テーブルＬのＬ［２］［‘ｓ’］には２（＝１＋０＋１）が入る。

【００４３】

ところで、図４及び図５を参照すると、テーブルＬにおいて、

Ｌ［ｉ］［ｃ＋１］＝Ｌ［ｉ］［ｃ］＋Ｆ［ｉ］［ｃ］－Ｆ［ｉ－１］［ｃ］

という関係がある。ただし、ｃ＋１は辞書的順序で文字ｃの次にくる文字である。また、Ｆ［０］［ｃ］＝０としている。例えば、上述したＬ［２］［‘ｓ’］の場合、

Ｌ［２］［‘ｓ’］
＝Ｌ［２］［‘ｐ’］＋Ｆ［２］［‘ｐ’］－Ｆ［１］［‘ｐ’］
＝１＋２－１＝２

となる。このことから、ｘ個おきの文字に対してのみテーブルＬを作成し、間の文字に対する値はテーブルＬ及びテーブルＦから算出することにより、メモリを節約することができる。ただし、この場合、この部分の計算時間はｘ倍となる。なお、図４の場合と同様に、テーブルＬにおいても‘＄’の列は省略することができる。

【００４４】

次に、テーブルＧを作成する。

上述したように、全てのｒ（０＜４＊ｒ＜ｎ＋４）に対して、Ｔ［ＳＡ［ｑ＋４＊（ｒ－１）］－１］（ただし、０＜ｑ≦４）の値が同じ物ごとに辞書的順序にしたがって並べ替えたものがテーブルＧ［ｒ］［１・・・４］である（ただし、ｒの値が上述した数１を満足する値である場合は０＜ｑ≦ｎ－（ｒ－１）＊４）。ここで、［ＳＡ［ｑ＋４＊（ｒ－１）］－１］は、配列Ｂにおいて、Ｂ［１・・・４］、Ｂ［５・・・８］、Ｂ［９・・・１２］に対応する。したがって、例えばＧ［１］［１・・・４］は、Ｂ［１・・・４］＝“ｓｓｍｐ”であるから対応するｑ＝１、２、３、４を‘ｓ’‘ｓ’‘ｍ’‘ｐ’の辞書的順序で並べ替えれば、

Ｇ［１］［１・・・４］＝｛３，４，１，２｝

となる（ｑ＝１の‘ｓ’とｑ＝２の‘ｓ’については、ｑの小さい方を先にしている）。同様に、ｒ＝２、３についても考え、結果として、

Ｇ［１・・・３］［１・・・４］
＝｛３，４，１，２｝、｛３，２，４，１｝、｛２，３，４，１｝

を得る。

【００４５】

次に、以上のテーブルＦ、Ｌ、Ｇを用いて計算されるｇ（ｉ，ｃ，ｊ）、ｆ’（ｊ，ｃ）及びｆ（ｊ，ｃ）について、具体的な算出例を挙げる。まず、ｇ（３，‘ｒ’，２）について、

ｇ（３，‘ｒ’，２）＝Ｇ［３］［Ｌ［３］［‘ｉ’］＋２］＋４＊（３－１）
＝Ｇ［３］［０＋２］＋８＝１１

となる。

また、ｆ’（１０，‘ｉ’）を求めるには、ｇ（３，‘ｉ’，ｘ）（０＜ｘ≦３）の中から１０以下の値を取る最大のｘを求めれば良い。上記と同様にｇ（３，‘ｒ’，１）、ｇ（３，‘ｒ’，３）を求めると、

ｇ（３，‘ｒ’，１）＝１０
ｇ（３，‘ｒ’，３）＝１２

であるから、ｆ’（１０，‘ｉ’）＝ｘ＝１が得られる。

さらに、ｆ（１０，‘ｉ’）の値は、

ｆ（１０，‘ｉ’）＝Ｆ［２］［‘ｉ’］＋ｆ’（１０，‘ｉ’）＝１＋１＝０

と求まる。

【００４６】

次に、上記のデータ構造を用いて、

Ｐ［１・・・３］＝“ｓｓｉ”

の検索を行う。

図３のフローチャートに示したアルゴリズムにおいて、まず、ｓｔａｒｔに

Ｃ［Ｐ［３］－１］＋１＝Ｃ［ｉ－１］＋１＝０＋１＝１

が代入され、ｅｎｄに

Ｃ［Ｐ［３］］＝Ｃ［ｉ］＝４

が代入される（ステップ３０１参照）。これは、検索のための中間パターンであるＰ［３］＝“ｉ”が、テキストＴに対する接尾辞配列ＳＡのどの範囲に位置しているかを示す。すなわち、ＳＡ［１・・・４］＝｛８　５　２　１１｝に対応するテキストＴの要素（テキストＴの８番目と５番目と２番目と１１番目の要素）が中間パターン“ｉ”と一致する。

【００４７】

次に、ｉ＝２（＝３－１）＞０であるので（ステップ３０１参照）、ｃにＰ［ｉ］が代入される（ステップ３０２、３０３参照）。そして、ｓｔａｒｔ及びｅｎｄに代入される値を計算する。すなわち、

ｓｔａｒｔ
＝Ｃ［Ｐ［２］－１］＋ｆ（ｓｔａｒｔ，Ｐ［２］）
＝Ｃ［‘ｓ’－１］＋ｆ（１，‘ｓ’）＝Ｃ［‘ｐ’］＋ｆ（１，‘ｓ’）

ここで、ｆ（１，‘ｓ’）は、ｋ＊ｉ＝１なのでテーブルＦから直接は求められず、

Ｃ［‘ｐ’］＋ｆ（１，‘ｓ’）
＝７＋Ｆ［１－１］［‘ｓ’］＋ｆ’（１，‘ｓ’）

ここで、Ｆ［０］［ｃ］＝０であり、ｆ’（１，‘ｓ’）は、ｇ（１，‘ｓ’，ｘ）でｘ＝１の時に

ｇ（１，‘ｓ’，１）
＝Ｇ［１］［Ｌ［１］［‘ｓ’］＋１］＋４＊（１－１）＝Ｇ［１］［２＋１］
＝１

となるので、ｆ’（１，‘ｓ’）＝１である。したがって、

Ｃ［‘ｐ’］＋ｆ（１，‘ｓ’）＝７＋０＋１＝８

となり、ｓｔａｒｔには８が代入される。また、

ｅｎｄ
＝Ｃ［Ｐ［２］－１］＋ｆ（ｅｎｄ，Ｐ［２］）
＝Ｃ［‘ｐ’］＋ｆ（４，‘ｓ’）

ここで、ｆ（４，‘ｓ’）は、ｋ＊ｉ＝４＊１なので、テーブルＦから直接求められ、Ｆ［１］［‘ｓ’］＝２であるから、

Ｃ［‘ｐ’］＋ｆ（４，‘ｓ’）＝７＋２＝９

となり、ｅｎｄには９が代入される。これは、検索のための中間パターンであるＰ［２　３］＝“ｓｉ”が、テキストＴに対する接尾辞配列ＳＡのどの範囲に位置しているかを示す。すなわち、ＳＡ［８］＝｛７｝とＳＡ［９］＝｛４｝とに対応するテキストＴの要素（テキストＴの７番目と４番目の要素）から始まる要素数２のパターンが中間パターン“ｓｉ”と一致する。

【００４８】

次に、ｅｎｄ（＝９）＞ｓｔａｒｔ（＝８）であるからステップ３０２に戻り

（ステップ３０４参照）、ｉ＝１（＝２－１）＞０であるので再度ステップ３０３に進み、ｃにＰ［ｉ］が代入される（ステップ３０２参照）。そして、ｓｔａｒｔ及びｅｎｄに代入される値を計算する。すなわち、

ｓｔａｒｔ
＝Ｃ［Ｐ［１］－１］＋ｆ（ｓｔａｒｔ，Ｐ［１］）
＝Ｃ［‘ｓ’－１］＋ｆ（８，‘ｓ’）＝Ｃ［‘ｐ’］＋ｆ（８，‘ｓ’）

ここで、ｆ（８，‘ｓ’）は、ｋ＊ｉ＝４＊２なので、テーブルＦから直接求められ、Ｆ［２］［‘ｓ’］＝３であるから、

Ｃ［‘ｐ’］＋ｆ（８，‘ｓ’）＝７＋３＝１０

となり、ｓｔａｒｔには１０が代入される。また、

ｅｎｄ
＝Ｃ［Ｐ［１］－１］＋ｆ（ｅｎｄ，Ｐ［１］）
＝Ｃ［‘ｐ’］＋ｆ（９，‘ｓ’）

ここで、ｆ（９，‘ｓ’）は、ｋ＊ｉ＝１１なのでテーブルＦから直接は求められず、４＊（ｉ－１）＜９＜＝４＊ｉからｉ＝３であるから、

Ｃ［‘ｐ’］＋ｆ（９，‘ｓ’）
＝７＋Ｆ［３－１］［‘ｓ’］＋ｆ’（９，‘ｓ’）

ここで、Ｆ［２］［‘ｓ’］はテーブルＦから３、ｆ’（９，‘ｓ’）は、ｇ（３，‘ｓ’，ｘ）でｘ＝１の時に

ｇ（３，‘ｓ’，１）

＝Ｇ［3］［Ｌ［3］［‘ｓ’］＋1］＋4＊（3－1）
＝Ｇ［3］［3＋1］＋8＝9

で、解はこれだけなので、ｆ’（9，‘ｓ’）＝ｘ＝１である。したがって、

Ｃ［‘ｐ’］＋ｆ（9，‘ｓ’）＝7＋3＋1＝１１

となり、ｅｎｄには１１が代入される。

【００４９】

次に、ｅｎｄ（＝１１）＞ｓｔａｒｔ（＝１０）であるからステップ３０２に戻り（ステップ３０４参照）、ｉ＝０（＝１－１）となったので（ステップ３０２参照）、ｓｔａｒｔ≦ｊ≦ｅｎｄであるような全てのｊに対して、ＳＡ［ｊ］の位置から始まるテキストＴの接尾辞を求める（ステップ３０６参照）。ここでは、ｓｔａｒｔ＝１０、ｅｎｄ＝１１であるから、ＳＡ［１０］＝6、ＳＡ［１１］＝3であり、Ｔ［3・・・5］＝Ｔ［6・・・8］＝“ｓｓｉ”となっており、パターンＰと一致している。

【００５０】

上記の動作例では、ｋ＝４である場合について説明したが、ｋの値は、検索対象であるテキストＴの文字数（ｎ）、検索パターンＰの文字数（ｍ）、テキストＴを構成するアルファベットにおける文字の種類の数（ｓ）などに応じて適宜に設定することができる。この場合、ｋの値に応じて、上述した前処理及び検索処理に必要なメインメモリ１０３の記憶容量とこれらの処理に要する時間とが変化する。大まかにはｋ＝Ｏ（ｓ）、すなわちｓの定数倍とすると、メインメモリ１０３に必要な記憶容量はＯ（ｎ log n）ビット、検索時間はＯ（ｍ log ｓ）となり、従来の接尾辞木を用いる検索方法における理論値と同じである。例えばｋ＝ｓのとき、３ｎ log ｓ＋２ｎ log ｎビットが必要となる。ただし、この場合が最小であるわけではない。

実際には、メインメモリ１０３の記憶容量は、８ビット、１６ビット、３２ビットの倍数（場合によっては約数）であることがほとんどなので、このことを考

慮してｋを設定することが好ましい。

【００５１】

次に、具体的なテキストＴに対して本実施の形態を適用した場合におけるメインメモリ１０３に必要な記憶容量（データサイズ）と検索時間とを例示する。

〔適用例１〕

文字が１バイトで表され、２５６種類である場合（終了判定文字＄も同時に表したい場合は２５５種類）。通常の英文テキストなどがこれに該当する。

この場合、テキストＴの文字数をｎとすれば、テキストＴのサイズはｎバイト、接尾辞配列ＳＡのサイズは４ｎバイトである。

例えば、ｋ＝６５５３６（＝2^{16}）とすると、ｋ以下の数字は２バイトで表すことができる。これにより、上述したテーブルＦ、Ｌ、Ｇ、Ｃの合計サイズは、２ｎバイト強となる。したがって、テキストＴ及び接尾辞配列ＳＡ、テキストＴを含んだデータサイズでも７ｎバイト強である。これは当該テキストＴに対する接尾辞木のサイズ（２０ｎ～４０ｎバイト程度）の３分の１程度である。

一方、検索速度は、log ｋに比例するので、ｋを小さくすると速度を上げることが可能である。

例えば、ｋ＝２５６（＝2^{8}）とすると、ｋ＝６５５３６の場合に対して２倍の検索速度を見込める。この場合、テーブルＦ、Ｌ、Ｇ、Ｃを持つために必要なメインメモリ１０３の記憶容量は６ｎバイトである。すなわち、テキストＴ及び接尾辞配列ＳＡを加えた総量でも１１ｎバイトのデータサイズとなり、やはり接尾辞木よりも小さい。

【００５２】

〔適用例２〕

文字が２バイトで表され、６５５３６（＝2^{16}）種類ある場合。日本語のテキストなどがこれに該当する。

この場合、ｋ＝６５５３６とすると、テーブルＦ、Ｌ、Ｇ、Ｃの合計サイズは、８ｎバイトであり、テキストＴ及び接尾辞配列ＳＡを加えた総量でも１４ｎバイトである。

なお、この例の場合、ｋ＝２５６などの小さい値とするのは、データサイズが

大きくなってしまうので現実的ではない。

【００５３】

〔適用例３〕

ＤＮＡの配列（文字の種類数は４）の場合。

２ｂｉｔの文字、４ｂｉｔの文字を扱うことを許すならば、ｋ＝４の場合、テーブルＦ、Ｌ、Ｇ、ＣとテキストＴ及び接尾辞配列ＳＡとを加えた総データサイズは８．７５ｎバイト程度となる。また、ｋ＝１６の場合、総データサイズは５．３７５ｎバイト程度となる。特に後者の場合、接尾辞配列ＳＡそのものとほとんど変わらないデータサイズとなっている。

【００５４】

次に、実際のＤＮＡ配列に対する検索速度の測定例を示す。

この測定例では、本実施の形態による検索方法と、接尾辞配列ＳＡを２分探索する従来の検索方法とを用いて、大腸菌の全配列に対し、同じクエリーを１００００００００回繰り返した場合の計算時間を比較している。なお、計算機は、ＣＰＵが３３３ＭＨｚＰｏｗｅｒ　ＰＣのＲＳ６０００（米国ＩＢＭ社のワークステーション）である。

検索パターンＰ＝“ＣＡＣＡＴＡＡ”
本実施の形態による検索時間　：０．３８秒
従来の２分探索による検索時間：４．３０秒

検索パターンＰ＝“ＡＧＡＧＣＧＧＣ”
本実施の形態による検索時間　：０．４７秒
従来の２分探索による検索時間：４．０２秒

検索パターンＰ＝“ＣＣＣＧＣＴＴＣＧＧＣ”
本実施の形態による検索時間　：０．７６秒
従来の２分探索による検索時間：３．３５秒

検索パターンＰ＝“ＡＣＣＧＣＧＡＡＡＴＡＣＣＧＧＣＧＣＧＧＡＡＡＴＣＡＴＣＧＡＣＴＴＡＣＧＣＡＴＡＧＧＣＧＣ”

本実施の形態による検索時間　：３．１３秒

従来の２分探索による検索時間：３．８８秒

検索パターンＰ＝“ＣＧＧＣＧＴＣＡＧＧＴＡＣＴＧＡＣＣＧＣＧＡＣＣＡＡＴＧＣＧＡ”

本実施の形態による検索時間　：０．８４秒

従来の２分探索による検索時間：３．４１秒

以上のように、全ての例において、本実施の形態の方が２分探索よりも計算時間が短縮（高速化）されている。最も高速化されている例（検索パターンＰ＝“ＡＧＡＧＣＧＧＣ”）では１０倍以上高速になっている。また、短い配列のクエリーほど高速化の効果があることがわかる。

【００５５】

なお、本実施の形態では、テキストＴの接尾辞配列ＳＡを探索して所望のパターンＰを検索する場合について説明したが、テキストＴの接頭辞配列を探索してパターンＰを検索することも可能である。

ここで、接頭辞とは、所定の文字列において、所定の文字を特定した場合の当該文字以前の文字列である。この接頭辞に対して、接尾辞に対する接尾辞木と同様の接頭辞木を生成することができる。また、接頭辞配列とは、テキストＴにおける全ての接頭辞を後から順に並べた文字列を、辞書的順序で並べ替えた場合のインデックスの配列である。すなわち、文字列の先頭方向と末尾方向（左右）を逆にしたテキストＴに対する接尾辞配列と本質的に同じである。したがって、方向を考慮することにより、上述した手法をそのまま接頭辞配列に対しても用いることができる。

【００５６】

【発明の効果】

以上説明したように、本発明によれば、大規模テキストデータベースの検索に

おいて、処理を行うためのデータ構造におけるデータサイズの増大を抑えながら、高速な検索を実現することができる。

【図面の簡単な説明】

【図１】　本実施の形態を実現するのに好適なコンピュータ装置のハードウェア構成の例を模式的に示した図である。

【図２】　本実施の形態におけるデータ構造の構築及び検索を行うための機能ブロックを示す図である。

【図３】　本実施の形態におけるパターンの検索アルゴリズムを説明するフローチャートである。

【図４】　本実施の形態において用いられるテーブルＦの構成例を示す図である。

【図５】　本実施の形態において用いられるテーブルＬの構成例を示す図である。

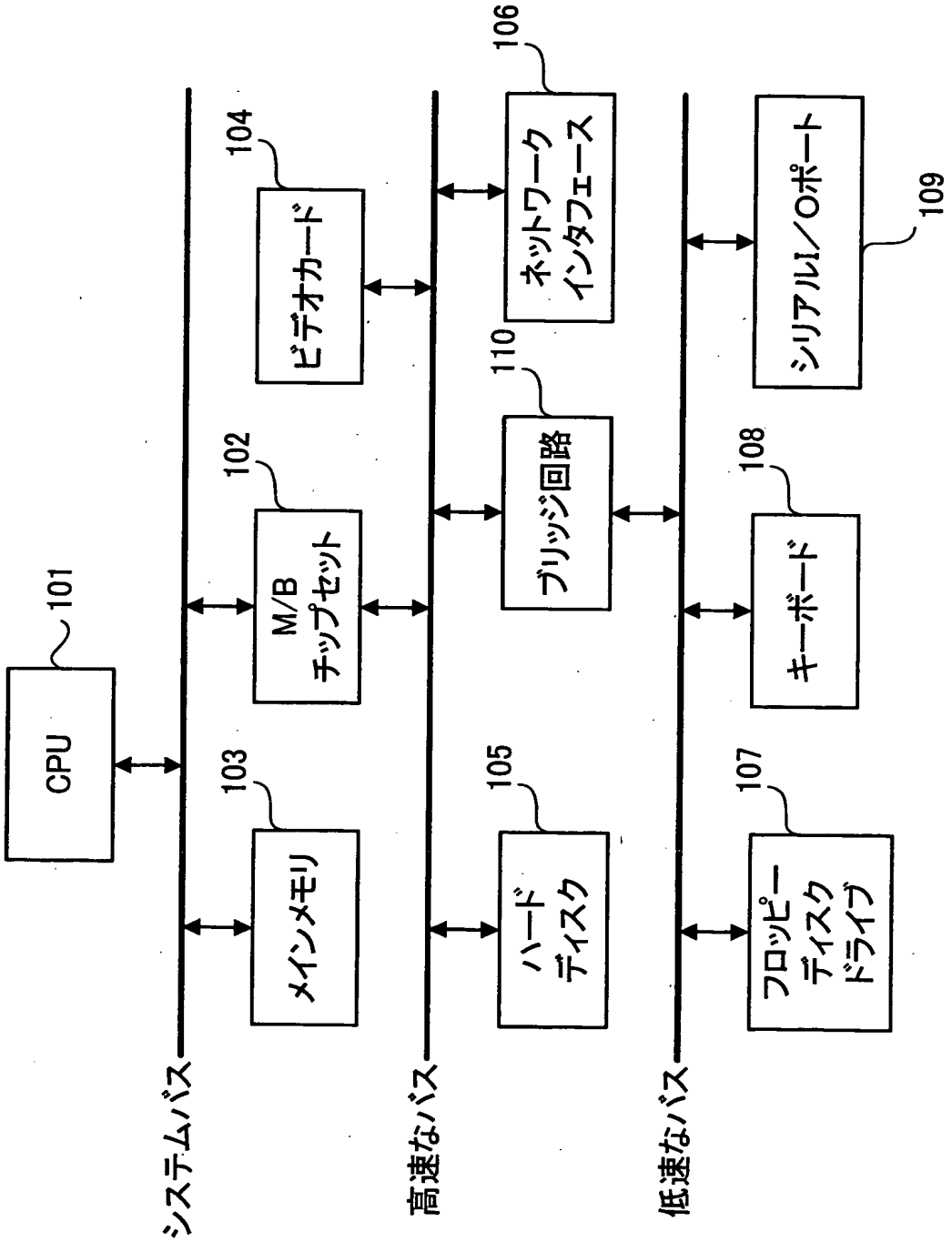
【図６】　接尾辞木の例を示す図である。

【符号の説明】

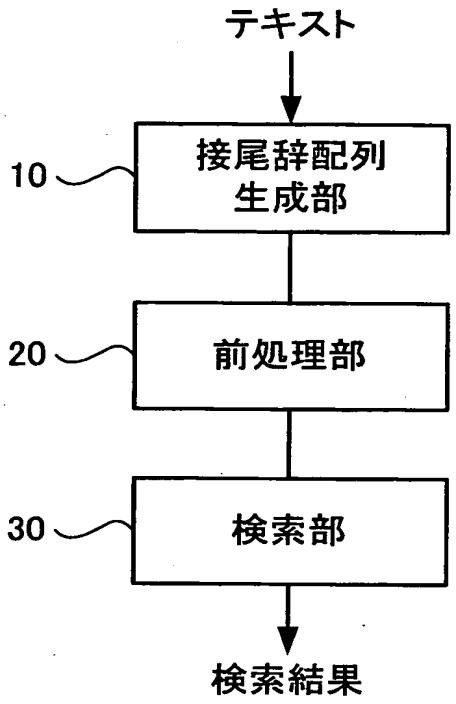
１０…接尾辞配列生成部、２０…前処理部、３０…検索部、１０１…ＣＰＵ（中央処理装置）、１０２…Ｍ／Ｂ（マザーボード）チップセット、１０３…メインメモリ、１０４…ビデオカード、１０５…ハードディスク、１０６…ネットワークインタフェース、１０７…フロッピーディスクドライブ、１０８…キーボード、１０９…シリアルＩ／Ｏポート、１１０…ブリッジ回路

【書類名】　　図面

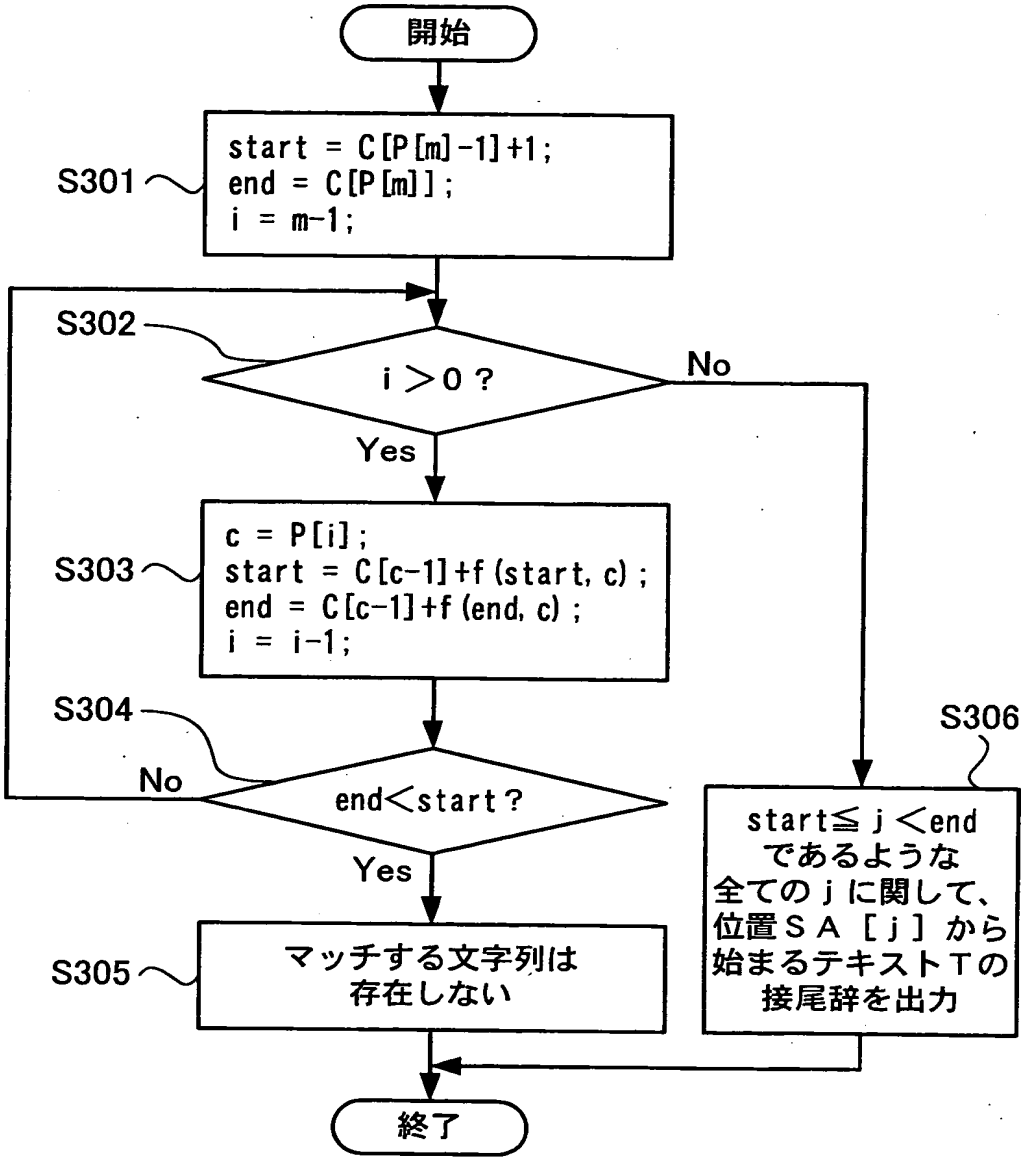
【図１】


CPU
101
M/B
チップセット
102
メインメモリ
103
ビデオカード
104
ハード
ディスク
105
ネットワーク
インタフェース
106
フロッピー
ディスク
ドライブ
107
ブリッジ回路
110
キーボード
108
シリアルI/Oポート
109
システムバス
高速なバス
低速なバス

【図２】



【図３】


開始
S301
start = C[P[m]-1]+1;
end = C[P[m]];
i = m-1;
S302
i＞0？
No
Yes
S303
c = P[i];
start = C[c-1]+f(start, c);
end = C[c-1]+f(end, c);
i = i-1;
S304
No
end＜start？
Yes
S305
マッチする文字列は
存在しない
S306
start≦ｊ＜end
であるような
全てのｊに関して、
位置ＳＡ［ｊ］から
始まるテキストＴの
接尾辞を出力
終了

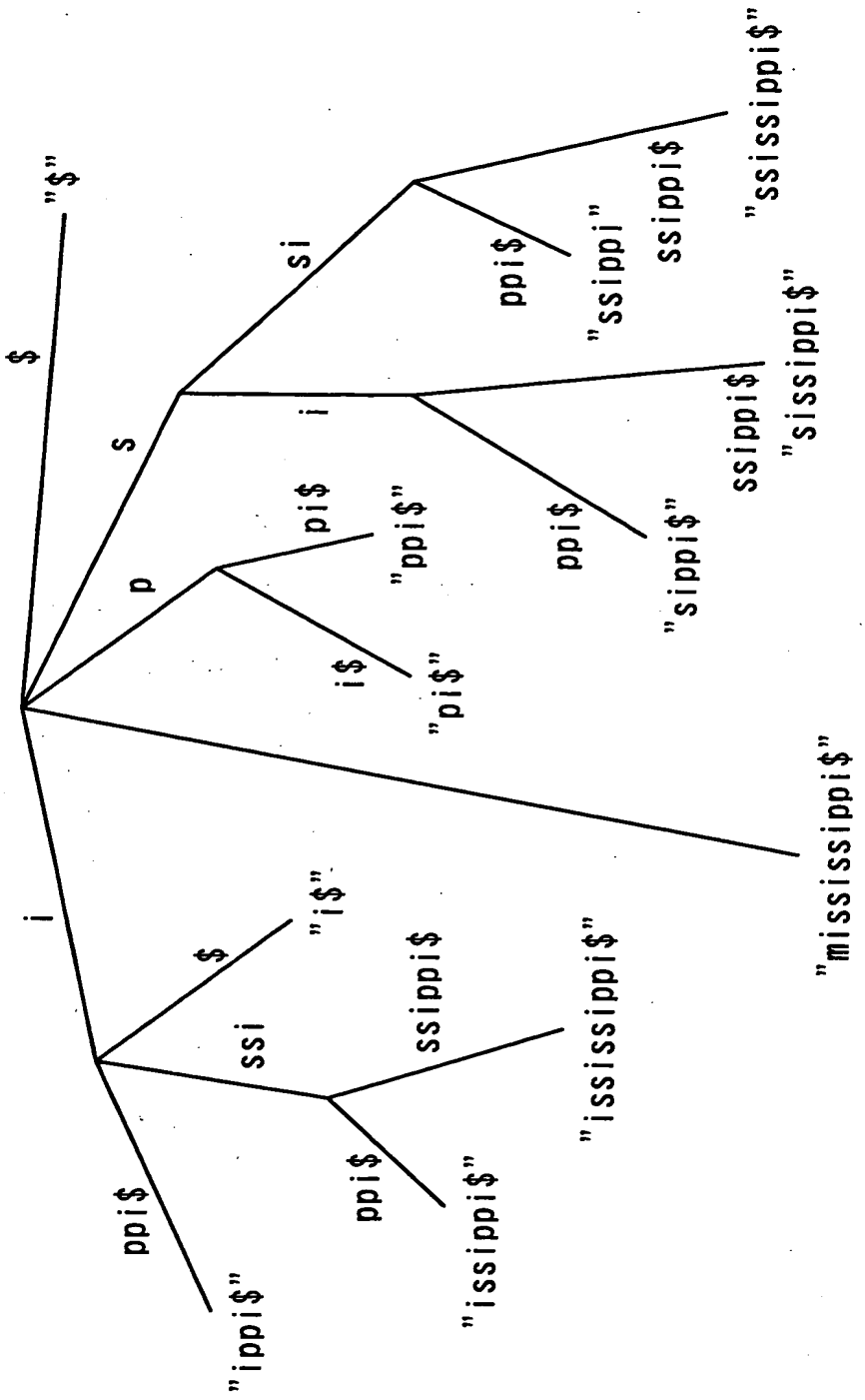
【図４】

i，c	i	m	p	s	$
1	0	1	1	2	0
2	1	1	2	3	1
3	4	1	2	4	1

【図５】

i，c	i	m	p	s	$
1	0	0	1	2	4
2	0	1	1	2	3
3	0	3	3	3	4

【図６】


"$"
$
s
si
ppi$
"ssippi"
ssippi$
"ssissippi$"
i
ssippi$
"sissippi$"
p
pi$
"ppi$"
ppi$
"sippi$"
i$
"pi$"
"mississippi$"
i
$
"i$"
ssi
ssippi$
"issississippi$"
ppi$
"issippi$"
ppi$
"ippi$"

【書類名】　　要約書

【要約】

【課題】　大規模テキストデータベースの検索において、処理を行うためのデータ構造におけるデータサイズの増大を抑えながら、高速な検索を実現する。

【解決手段】　検索対象である文字列中から所望のパターンを検索するパターン検索方法において、次の範囲検索ステップと、文字列抽出ステップとを含む。すなわち、範囲検索ステップにおいて、このパターンの最後の文字から前方へ１文字ずつ順に加えて得られる各中間パターンに関して、この中間パターンの先頭の文字が検索対象の文字列に対する接尾辞配列のどの範囲に存在するかを順次検索する。次に、文字列抽出ステップにおいて、当該接尾辞配列の範囲に含まれる各要素に対応する文字列の要素を特定し、この文字列の各要素を先頭としてこのパターンの要素数と同じ数の要素からなる部分文字列を抽出する文字列抽出ステップとを含むことを特徴とする。

【選択図】　　なし

認定・付加情報

特許出願の番号	特願２００１－００４１８９
受付番号	５０１００００２９４１１
書類名	特許願
担当官	末武　実　　１９１２
作成日	平成１３年　２月２７日

＜認定情報・付加情報＞

【特許出願人】
- 【識別番号】 390009531
- 【住所又は居所】 アメリカ合衆国１０５０４、ニューヨーク州　アーモンク　（番地なし）
- 【氏名又は名称】 インターナショナル・ビジネス・マシーンズ・コーポレーション

【代理人】
- 【識別番号】 100086243
- 【住所又は居所】 神奈川県大和市下鶴間１６２３番地１４　日本アイ・ビー・エム株式会社　大和事業所内
- 【氏名又は名称】 坂口　博

【代理人】
- 【識別番号】 100091568
- 【住所又は居所】 神奈川県大和市下鶴間１６２３番地１４　日本アイ・ビー・エム株式会社　大和事業所内
- 【氏名又は名称】 市位　嘉宏

【代理人】
- 【識別番号】 100106699
- 【住所又は居所】 神奈川県大和市下鶴間１６２３番１４　日本アイ・ビー・エム株式会社大和事業所内
- 【氏名又は名称】 渡部　弘道

【復代理人】 申請人
- 【識別番号】 100104880
- 【住所又は居所】 東京都港区赤坂５－４－１１　山口建設第２ビル６Ｆ　セリオ国際特許事務所
- 【氏名又は名称】 古部　次郎

【選任した復代理人】
- 【識別番号】 100100077

次頁有

認定・付加情報（続き）

【住所又は居所】　東京都港区赤坂５－４－１１　山口建設第２ビル
６Ｆ　セリオ国際特許事務所
【氏名又は名称】　大場　充

次頁無

出　願　人　履　歴　情　報

識別番号　　〔３９０００９５３１〕

１．変更年月日　　２０００年　５月１６日
［変更理由］　名称変更
住　所　　アメリカ合衆国１０５０４、ニューヨーク州　アーモンク　（番地なし）
氏　名　　インターナショナル・ビジネス・マシーンズ・コーポレーション